Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information and Computation 198 (2005) 1–23
Information and Computation
www.elsevier.com/locate/ic

Competing provers yield improved Karp–Lipton collapse results☆

Jin-Yi Cai [a,c], Venkatesan T. Chakaravarthy [a], Lane A. Hemaspaandra [b,*], Mitsunori Ogihara [b]

[a] *Computer Sciences Department, University of Wisconsin, Madison, WI 53706, USA*
[b] *Department of Computer Science, University of Rochester, Rochester, NY 14627, USA*
[c] *Department of Computer Science, Tsinghua University, Beijing, China*

Received 22 May 2003; revised 15 September 2004
Available online 7 March 2005

Abstract

Via competing provers, we show that if a language A is self-reducible and has polynomial-size circuits then $S_2^A = S_2$. Building on this, we strengthen the Kämper–AFK theorem, namely, we prove that if NP $\subseteq$ (NP $\cap$ coNP)/poly then the polynomial hierarchy collapses to $S_2^{NP \cap coNP}$. We also strengthen Yap's theorem, namely, we prove that if NP $\subseteq$ coNP/poly then the polynomial hierarchy collapses to S_2^{NP}. Under the same assumptions, the best previously known collapses were to ZPP^{NP} and $ZPP^{NP^{NP}}$, respectively ([SIAM Journal on Computing 28 (1) (1998) 311; Journal of Computer and System Sciences 52 (3) (1996) 421], building on [Proceedings of the 12th ACM Symposium on Theory of Computing, ACM Press, New York, 1980, pp. 302–309; Journal of Computer and System Sciences 39 (1989) 21; Theoretical Computer Science 85 (2) (1991) 305; Theoretical Computer Science 26 (3) (1983) 287]). It is known that $S_2 \subseteq ZPP^{NP}$ [Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Silver Spring, MD, 2001, pp. 620–629]. That result and its relativized version show that our new collapses indeed improve the

☆ Supported in part by NIH Grants RO1-AG18231 and P30-AG18254, and NSF Grants INT-9726724, CCR-9701911, INT-9815095, DUE-9980943, EIA-0080124, CCR-0196197, EIA-0205061, and NSF-CCF-0426761. A preliminary version appeared in the STACS '03 conference [14].

* Corresponding author. Fax: +1 585 461 4556.
E-mail addresses: jyc@cs.wisc.edu (J. Cai), venkat@cs.wisc.edu (V.T. Chakaravarthy), lane@cs.rochester.edu (L.A. Hemaspaandra), ogihara@cs.rochester.edu (M. Ogihara).

doi:10.1016/j.ic.2005.01.002

previously known results. The Kämper–AFK theorem and Yap's theorem are used in the literature as bridges in a variety of results—ranging from the study of unique solutions to issues of approximation—and so our results implicitly strengthen those results.

Keywords: Structural complexity; Competing provers; Nonuniform complexity; Symmetric alternation; Karp–Lipton theorem; Yap's theorem; Kämper–AFK theorem; Lowness

1. Proving collapses via competing provers

The symmetric alternation class S_2 was introduced by Canetti [16] and Russell and Sundaram [40]. In one model that captures this notion, we have two all-powerful competing provers, the Yes-prover and the No-prover, and a polynomial-time verifier. Given an input string x, the Yes-prover and the No-prover attempt to convince the verifier of $x \in L$ and of $x \notin L$, respectively. To do so, they provide proofs (i.e., bitstrings) y and z, respectively. Then the verifier simply checks whether y is a correct (in whatever sense of "correct" that the verifier happens to enforce) one for $x \in L$ and whether z is a correct one for $x \notin L$, and votes in favor of one of the provers. We require that if $x \in L$ then the Yes-prover has an irrefutable proof y that can withstand any challenge z from the No-prover; and if $x \notin L$ then the No-prover has an irrefutable proof z that can withstand any challenge y from the Yes-prover. Languages with such a proof system are said to be in the class S_2. We define the class formally in Section 3.

When we allow the verifier to have access to an oracle A, we obtain the relativized class S_2^A. Our main result gives a partial characterization for sets that are not useful as oracles to the verifier:

Theorem: *If A is self-reducible and has polynomial-size circuits then*

$$S_2^A = S_2.$$

We note that similar results are known for NP^{NP} [8] and ZPP^{NP} [33]. The above result is useful in obtaining a number of conditional collapse results. For example, we can show that if NP has polynomial-size circuits then the polynomial hierarchy, PH, collapses to S_2. This follows from the fact that SAT is a self-reducible many-one complete problem for NP. Though this result is already known (see [13]), our result provides a general method to obtain conditional collapses for other classes. We can apply the theorem to other complexity classes with a set of self-reducible languages that are "collectively" many-one complete for the class (e.g., UP, FewP, NP, Σ_k^p, $\oplus$P). Moreover, by using a relativized version of the above theorem, we can obtain collapses for the first time to $S_2^{NP \cap coNP}$ and S_2^{NP} (under assumptions weaker than those yielding collapses to S_2). For example, we will show that (i) if $NP \subseteq (NP \cap coNP)/poly$ then PH collapses to $S_2^{NP \cap coNP}$; (ii) if $NP \subseteq coNP/poly$ then PH collapses to S_2^{NP}. Previously the best known collapse results under the same assumptions were to ZPP^{NP} and $ZPP^{NP^{NP}}$ ([11,33], building on [1,28,29,53]). Since $S_2 \subseteq ZPP^{NP}$ [13] (and because this result relativizes), we see that the new collapses are indeed improvements. In Section 2, we discuss the motivation behind the theorem in more detail.

We introduce and use the technique of a "dynamic contest" to prove the above theorem. The first hurdle is to show that if A is self-reducible and has polynomial-size circuits then $A \in S_2$. Upon input

x, suppose each prover provides a circuit of appropriate size to the verifier. Of course, the honest prover can provide the correct circuit. And by simulating the circuit with x as input, the verifier can determine the membership of x in A correctly. But the issue is that the polynomial-time verifier needs to first find out which one of the two circuits is the correct one! The idea is to simulate the circuits on a sequence of successively smaller strings. The strings are chosen dynamically, using the self-reducing algorithm of A and the outputs of the circuits on earlier strings in the sequence. Using this idea of a dynamic contest between the circuits, we show how the verifier can always choose a correct circuit (if at least one of the two is correct). Then the honest prover can provide the correct circuit and win the vote, irrespective of the circuit provided by the other prover. We then extend the proof to show that $S_2^A = S_2$. The details of this proof can be found in Section 4.

2. Background and motivation

Karp and Lipton [29], who in their work credit an important contribution by Sipser, proved that if $NP \subseteq P/poly$ (equivalently, if some sparse set is Turing-hard for NP) then $NP^{NP} = PH$. (For more on P/poly and other consequences of $NP \subseteq P/poly$, see, e.g. [4,31].) Köbler and Watanabe [33] (see also Bshouty et al. [11]) strengthened this result by showing that if $NP \subseteq P/poly$ then $ZPP^{NP} = PH$. It is known ([3], see also [27]) that there are relativized worlds in which $NP \subseteq P/poly$ does not imply the collapse of the boolean hierarchy (and so certainly does not imply $P = NP$).

The just-mentioned Köbler–Watanabe result has itself been further strengthened, via the combination of two results of independent importance: First, Sengupta (see [13]) observed that an alternative proof of the Karp–Lipton theorem by Hopcroft [26] in fact shows that $NP \subseteq P/poly$ implies $S_2 = PH$, where S_2 is the symmetric alternation class of Canetti [16] and Russell and Sundaram [40]. Second, Cai [13] proved that $S_2 \subseteq ZPP^{NP}$, thus showing that the Hopcroft–Sengupta collapse of PH to S_2 is at least as deep a collapse as that of Bshouty et al. and Köbler–Watanabe. Currently this is the strongest form of the Karp–Lipton theorem.

The Karp–Lipton result and the Köbler–Watanabe result have been generalized to "lowness" results. Regarding the former, we have for example the lowness result of Balcázar et al. ([8], see also [7,35]) that every Turing self-reducible set A in P/poly is low for NP^{NP}, i.e., $NP^{NP^A} = NP^{NP}$. Regarding the latter, Köbler and Watanabe [33] themselves proved that every Turing self-reducible set A in $(NP \cap coNP)/poly$ is low for ZPP^{NP}, i.e., $ZPP^{NP^A} = ZPP^{NP}$.

Why might one wish to make such transitions from conditional collapse results? A first reason is aesthetic and philosophical. Though conditional collapse results such as "$NP \subseteq P/poly \Longrightarrow NP^{NP} = PH$" are certainly valuable (in fact, we will speak below about how lowness results flexibly unify and yield conditional collapse results), note that the related lowness result, "every Turing self-reducible set in P/poly is low for NP^{NP}," not only has the practical merit of (as can be shown) implying the former result, but also proves, unconditionally, that a class of sets (the class of all Turing self-reducible sets in P/poly) exhibits a simplicity property (namely, giving absolutely no additional power to NP^{NP} when used as free information). A second reason why making the transition from conditional collapse results to lowness results can be valuable is directly utilitarian. Lowness results are generally a more broadly applicable tool in obtaining collapse consequences for a wide variety of classes. In practice, lowness results in our settings will often apply crisply and directly to all classes having Turing-self-reducible many-one complete sets, and even to all classes $\mathcal{C}$ having a set

of self-reducible languages that are "collectively" many-one complete for the class. Among natural classes having such properties are UP, FewP, NP, coUP, coFewP, coNP, Σ_k^p, Π_k^p, $\oplus$P, and PSPACE. Even if it is possible to prove these results for each class separately, it is much more desirable to have a single proof. Making a transition from conditional collapse results to lowness results has been previously achieved for the Karp–Lipton Σ_2^p result and for Köbler–Watanabe $\mathrm{ZPP}^{\mathrm{NP}}$ result. The present paper essentially achieves this transition for the Hopcroft–Sengupta S_2 result.

We say "essentially" since S_2 presents strong barriers to the transition—barriers that are not present, even by analogy, for the $\mathrm{NP}^{\mathrm{NP}}$ and $\mathrm{ZPP}^{\mathrm{NP}}$ cases. The source of the barrier is deeply ingrained in the nature of S_2. While $\mathrm{NP}^{\mathrm{NP}}$ and $\mathrm{ZPP}^{\mathrm{NP}}$ both have as their "upper level" an unfettered access to existential quantification, S_2 by its very definition possesses a quite subtly constrained quantification structure. For the case of P/poly, this problem does not affect us, and we are able to establish the following lowness result: *All Turing self-reducible sets in* P/poly *are low for* S_2. However, for the case of (NP ∩ coNP)/poly the restrictive structure of S_2 is remarkably hostile to obtaining pure lowness results. Nonetheless, we obtain the following lowness-like result that we show is useful in a broad range of new, strongest-known collapses: *For all Turing self-reducible sets A in* (NP ∩ coNP)/poly *and all sets B that are Turing reducible to A* (or even $\leq_T^{rs}$-reducible to *A*), $\mathrm{S}_2^B \subseteq \mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$.

In showing that the above lowness and lowness-like results do yield conditional collapse consequences, it will be important to know that Cai's $\mathrm{S}_2 \subseteq \mathrm{ZPP}^{\mathrm{NP}}$ result relativizes, and we note that it does. We also establish (as Theorem 5.6) that the Hopcroft–Sengupta result relativizes flexibly.

So, putting this all together, we establish a collection of lowness results and tools that allow, for a very broad range of classes, strong uniform-class consequences to be read off from assumed containments in nonuniform classes. The most central of these results is that we strengthen the Kämper–AFK theorem (which says that $\mathrm{NP} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly} \Longrightarrow \mathrm{PH} = \mathrm{NP}^{\mathrm{NP}}$), relative to both its just-mentioned original version [1,28] and the strengthened version due to Köbler–Watanabe [33]. In particular, we prove that $\mathrm{NP} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly} \Longrightarrow \mathrm{PH} = \mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$. Another central result we strengthen is Yap's theorem. Yap's theorem [53] states that if $\mathrm{NP} \subseteq \mathrm{coNP}/\mathrm{poly}$ (or, equivalently, if $\mathrm{coNP} \subseteq \mathrm{NP}/\mathrm{poly}$) then $\mathrm{PH} = \Sigma_3^p$. Köbler and Watanabe strengthened Yap's theorem by showing that if $\mathrm{NP} \subseteq \mathrm{coNP}/\mathrm{poly}$ then $\mathrm{PH} = \mathrm{ZPP}^{\Sigma_2^p}$ [33]. We further strengthen Yap's theorem, via showing that if $\mathrm{NP} \subseteq \mathrm{coNP}/\mathrm{poly}$ then $\mathrm{PH} = \mathrm{S}_2^{\mathrm{NP}}$. (Note: $\mathrm{S}_2^{\mathrm{NP}} \subseteq \mathrm{ZPP}^{\Sigma_2^p} \subseteq \Sigma_3^p$.)

This paper explores the relationship between (NP ∩ coNP)/poly and classes up to and including PSPACE. Regarding the relationship between (NP ∩ coNP)/poly and classes beyond PSPACE, we commend to the reader the work of Variyam [49] and Vinodchandran [50], who shows for example that an exponential-time analog of AM is not contained in (NP ∩ coNP)/poly.

The paper is organized as follows. Section 3 presents definitions. Section 4 proves our main lowness theorems about S_2. In Section 5 we use our lowness theorems, in combination with a close look at interactive provers, to obtain collapse results for many complexity classes. Section 6 presents some open questions.

3. Definitions

In this section we present the required definitions and notations. Throughout this paper all polynomials are without negative coefficients so they are monotonically nondecreasing and for all $n \geqslant 0$

their values at n are nonnegative. Throughout this paper, $(\exists^m y)$ will denote $(\exists y : |y| = m)$, and $(\forall^m y)$ will denote $(\forall y : |y| = m)$.

We now define the symmetric alternation class S_2.

Definition 3.1 ([16,40])**.** A language L is in S_2 if there exists a polynomial-time computable 3-argument boolean predicate P and a polynomial p such that, for all x,

(1) $x \in L \iff (\exists^{p(|x|)} y)(\forall^{p(|x|)} z)[P(x, y, z) = 1]$, and
(2) $x \notin L \iff (\exists^{p(|x|)} z)(\forall^{p(|x|)} y)[P(x, y, z) = 0]$.

(In literature, the class denoted S_2 above has had other notations, such as S_2^p and S_2^P.) Now that the reader has seen this formal definition, we repeat the interpretation commented on earlier (with a bit more fine print). Namely, we can interpret the above definition as follows. Suppose there are two competing all-powerful provers, the Yes-prover and the No-prover, interacting with a polynomial-time verifier, P. Given an input string x, the Yes-prover and the No-prover attempt to convince the verifier of $x \in L$ and of $x \notin L$, respectively. To do so, they provide "proofs" y and z, respectively. Then the verifier simply checks whether y is a "correct" one for $x \in L$ and whether z is a "correct" one for $x \notin L$. If $x \in L$ then the Yes-prover has an irrefutable proof y that can withstand any (correct-length[1]) challenge z from the No-prover; and if $x \notin L$ then the No-prover has an irrefutable proof z that can withstand any (correct-length) challenge y from the Yes-prover.

Since relativizing S_2 will be important in this paper, we generalize S_2 in a flexible way that allows us to rigorously specify what we mean by relativized S_2, and that also potentially itself opens the door to the study of S_2-like notions applied to a wide variety of classes of predicates.

Definition 3.2. Let $\mathcal{C}$ be any complexity class. We define $S_2[\mathcal{C}]$ to be the class of all sets L such that there exists a 3-argument boolean predicate $P \in \mathcal{C}$ and a polynomial q such that

(1) $x \in L \iff (\exists^{q(|x|)} y)(\forall^{q(|x|)} z)[P(x, y, z) = 1]$, and
(2) $x \notin L \iff (\exists^{q(|x|)} z)(\forall^{q(|x|)} y)[P(x, y, z) = 0]$.

One might alternatively use a pairing function to pair $\langle x, y, z\rangle$. For a nice class $\mathcal{C}$, all choices of pairing functions having the standard properties of pairing functions would yield the same class as each other. However, pathological classes $\mathcal{C}$ (e.g., classes containing a single set) would be sensitive to the choice of pairing function. Also, we have followed the Russell–Sundaram feature of using the same polynomial bounds on y and z. Again, for nice classes $\mathcal{C}$ this yields the same class $S_2[\mathcal{C}]$ as if we had allowed separate polynomials. However, for pathological classes $\mathcal{C}$ the one- and two-polynomial approaches will change the class defined.

We now define our relativizations of S_2 that give the P-time predicate of S_2 access to an oracle.

[1] If we wished to, we could alter the verifier in such a way as to remove the "correct-length" restriction on this claim, since if one modifies the definition of S_2 by replacing $(\exists^{p(|x|)} y)(\forall^{p(|x|)} z)$ with $(\exists^{p(|x|)} y)(\forall z)$ and by replacing $(\exists^{p(|x|)} z)(\forall^{p(|x|)} y)$ with $(\exists^{p(|x|)} z)(\forall y)$, the overall class defined is unchanged.

Definition 3.3.

(1) For each set A, S_2^A denotes $S_2[P^A]$.
(2) For each class $\mathcal{C}$, $S_2^{\mathcal{C}}$ denotes $\bigcup_{A\in\mathcal{C}} S_2^A$.

Definition 3.4 (*see* [18,34])**.** We say that a nondeterministic polynomial-time Turing machine (NPTM) M is *strong with respect to* (*an oracle*) B, if for all inputs x, $M^B(x)$ satisfies: (i) each computation path halts in one of the states *accept*, *reject*, or "?", (ii) if there is an accepting path there are no rejecting paths, and (iii) if there is a rejecting path there are no accepting paths, and (iv) at least one path accepts or rejects. M is said to be *robustly strong* if M is strong with respect to every oracle B. The language defined by M^B is the set of all strings x such that $M^B(x)$ has at least one accepting path.

Proposition 3.5 ([34] see also [41, 43])**.** *A language A is in* $NP^B \cap coNP^B$ *if and only if there is a NPTM M such that M is strong with respect to B and $A = L(M^B)$. In particular, a set A belongs to* NP ∩ coNP *if and only if there is a NPTM M strong with respect to $\emptyset$ such that $A = L(M)$.*

Definition 3.6 ([18])**.** We say that $B \leq_T^{rs} A$, if there exists a robustly strong NPTM M such that $B = L(M^A)$.

For each a and b such that $\leqslant_b^a$ is a defined reduction, and for each class $\mathcal{C}$, $R_a^b(\mathcal{C})$ will denote $\{B|(\exists A \in \mathcal{C})[B \leqslant_a^b A]\}$. We say that a class $\mathcal{C}$ is *closed under polynomial-time many-one reductions* if $\{L \mid (\exists C \in \mathcal{C})[L \leq_m^p C]\} \subseteq \mathcal{C}$.

In the following definition and in invocations of it, $\langle\cdot,\cdot\rangle$ denotes some fixed, standard pairing function (having the standard nice properties such as injectivity, surjectivity, polynomial-time computability, and polynomial-time invertibility).

Definition 3.7 ([29])**.** Let $\mathcal{C}$ be a complexity class. A language L is said to be in $\mathcal{C}$/poly if there exist a language $L' \in \mathcal{C}$, a function s, and a polynomial p for which the following conditions hold:

(1) For all $n \geqslant 0$, $s(1^n)$ is a string bounded in length by $p(n)$.
(2) For all x, $x \in L \iff \langle x, s(1^{|x|})\rangle \in L'$.

Definition 3.8 (*see the survey* [32])**.** For each class $\mathcal{C}$ for which relativization is well defined and for each set A, we say that A is low for $\mathcal{C}$ if $\mathcal{C}^A = \mathcal{C}$. For a class $\mathcal{D}$, we say that $\mathcal{D}$ is low for $\mathcal{C}$ exactly if each set in $\mathcal{D}$ is low for $\mathcal{C}$.

Self-reducibility (sometimes called downward self-reducibility) is a central, widely used concept in complexity theory, and will be important in this paper.[2]

[2] Definition 3.9 is a very natural and widely used one ([9], see also [38]). We mention, however, that some authors prefer to define Turing self-reducibility not in this length-based way, but rather in terms of allowing general orderings satisfying appropriate polynomial-time computability requirements and having the property that there is a polynomial q such that, for all x, all strictly descending chains from x are of length at most $q(|x|)$ bits (see [30,37] for full details). We note that every claim/theorem made in this paper for Definition 3.9's length-based notion of self-reducibility also holds under such "nice-p-order"-based definition of Turing self-reducibility.

Definition 3.9 (*see, e.g.* [9])**.** A set B is said to be *Turing self-reducible* if there is a (clocked, deterministic) polynomial-time Turing machine M that has the following two properties:

(1) $B = L(M^B)$.
(2) On each input string x, regardless of the answers it receives to oracle queries, M never queries any string of length greater than or equal to $|x|$.

Definition 3.10.

(1) For sets B and C, we say that B *is Turing self-reducible with respect to* C if there is a (clocked, deterministic) polynomial-time Turing machine M such that the following properties hold.
   (a) $B = L(M^{B,C})$, where in this model M has both an oracle tape for querying B and a separate oracle tape for querying C.
   (b) On each input string x, regardless of the answers it receives to previous oracle queries from either of its oracles, M never queries on its oracle tape corresponding to B any string of length greater than or equal to $|x|$.
(2) For a set B and a class $\mathcal{C}$, we say that B *is Turing self-reducible with respect to* $\mathcal{C}$ if there is a set $C \in \mathcal{C}$ such that B is Turing self-reducible with respect to C.

All complexity classes (e.g., NP, coNP, ZPP, $\mathrm{Mod}_k\mathrm{P}$, PSPACE, etc.) have their standard definitions (see [25]). For clarity, we mention explicitly that, as is standard, $\mathrm{Mod}_k\mathrm{P}$ is the class of all sets A such that for some nondeterministic polynomial-time machine M, and each x, it holds that $x \in A$ if and only if the number of accepting paths of $M(x)$ is not congruent to 0 mod k.

4. Lowness results for S_2

In this section, we establish some lowness results about S_2 and $\mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$. We also prove some lowness transference lemmas. We use these results in Section 5 to obtain collapse results for many complexity classes.

Theorem 4.1. *If* $A \in \mathrm{P/poly}$ *and* A *is Turing self-reducible then* A *is low for* S_2, *i.e.*, $\mathrm{S}_2^A = \mathrm{S}_2$.

Proof. Let A be as in the hypothesis of the theorem. Let L be an arbitrary language in S_2^A. There exist a 3-argument predicate $B \in \mathrm{P}^A$ and a polynomial p such that, for all x, (i) $x \in L \iff (\exists^{p(|x|)} y)(\forall^{p(|x|)} z)[B(x,y,z) = 1]$, and (ii) $x \notin L \iff (\exists^{p(|x|)} z)(\forall^{p(|x|)} y)[B(x,y,z) = 0]$. Since $B \in \mathrm{P}^A$ there exists a polynomial-time oracle Turing machine M_0 that decides B with oracle A. Let q be a polynomial such that, for all x, y, and z satisfying $|y| = |z| = p(|x|)$, all query strings of M_0 on input $\langle x, y, z\rangle$ have length at most $q(|x|)$ irrespective of its oracle. Since $A \in \mathrm{P/poly}$, there exist a language in P and an advice function s witnessing that $A \in \mathrm{P/poly}$. Let S be the function such that, for all x, $S(x) = s(1^0)\#s(1)\#\cdots\#s(1^{|x|})$, where # is a delimiter. Then there exists a polynomial r such that S is polynomially length-bounded by r and there exists a polynomial-time machine M_{adv} such that for all x and n, $n \geqslant |x|$, M_{adv} on $\langle x, S(1^n)\rangle$ correctly decides whether $x \in A$. (This is what is sometimes called in the literature "strong advice"—advice that works not just at one length but also on all strings up

to that given length, see [10].) Since A is Turing self-reducible, there exists a polynomial-time oracle Turing machine $M_{\rm sr}$ such that $M_{\rm sr}$, given A as its oracle, correctly decides A, and such that, for all x, irrespective of the oracle, every query string (if any) of $M_{\rm sr}$ on input x has length strictly less than $|x|$.

We first define a deterministic polynomial-time machine that will be called the *A-simulator*, which will take as its input a triple $\langle w, s_1, s_2\rangle$ and then will output either $M_{\rm adv}(\langle w, s_1\rangle)$ or $M_{\rm adv}(\langle w, s_2\rangle)$. On input $\langle w, s_1, s_2\rangle$, the A-simulator will first compute $M_{\rm adv}(\langle w, s_1\rangle)$ and $M_{\rm adv}(\langle w, s_2\rangle)$. If they agree on their outcome, the A-simulator outputs that outcome and halts. Otherwise, it sets a variable α to w and simulates $M_{\rm sr}(\alpha)$ answering its queries β by running both $M_{\rm adv}(\langle \beta, s_1\rangle)$ and $M_{\rm adv}(\langle \beta, s_2\rangle)$ as long as they agree. If for some query β, $M_{\rm adv}(\langle \beta, s_1\rangle)$ and $M_{\rm adv}(\langle \beta, s_2\rangle)$ disagree, then the A-simulator sets α to β and starts over the above procedure. Since $M_{\rm sr}$ is a self-reduction, the queries are always shorter than the input, so the length of α becomes smaller on each iteration. Thus, if the "otherwise" case above was reached, then there is eventually a point at which α satisfies (i) $M_{\rm adv}(\langle \alpha, s_1\rangle)$ and $M_{\rm adv}(\langle \alpha, s_2\rangle)$ disagree and (ii) for all queries β of $M_{\rm sr}$ on input α, $M_{\rm adv}(\langle \beta, s_1\rangle)$ and $M_{\rm adv}(\langle \beta, s_2\rangle)$ agree. For such an α, there is exactly one $t \in \{s_1, s_2\}$ such that $M_{\rm adv}(\langle \alpha, t\rangle)$ agrees with $M_{\rm sr}(\alpha)$ when all the queries are answered by $M_{\rm adv}$ with t as the advice string (note that, on these particular queries made by $M_{\rm sr}$, using s_1 as the advice string and using s_2 as the advice string produce the exact same answers). The A-simulator finds which of s_1 and s_2 is this t and outputs the value of $M_{\rm adv}(\langle w, t\rangle)$. Since the length of α decreases each iteration and both $M_{\rm adv}$ and $M_{\rm sr}$ are polynomial time-bounded, the A-simulator runs in polynomial time.

We show that L is in S_2 by developing its verification scheme with the Yes- and No-provers as discussed in the paragraph after Definition 3.1. Let x be an input. The Yes-prover's certificate Y and the No-prover's certificate Z are of the form $\langle y, s_1\rangle$ and $\langle z, s_2\rangle$, respectively, where $|y| = |z| = p(|x|)$ and $|s_1|, |s_2| \leqslant r(q(|x|))$. The verifier attempts to evaluate $B(x, y, z)$ using s_1 and s_2. To do this, the verifier simulates M_0 on input $\langle x, y, z\rangle$. When M_0 makes a query, say w, the verifier computes the answer from the oracle by running the A-simulator on input $\langle w, s_1, s_2\rangle$. The verification process clearly runs in polynomial time.

Suppose $x \in L$. Take the string y to be such that for all z satisfying $|z| = p(|x|)$ it holds that $B(x, y, z) = 1$, and take s_1 to be $S(1^{q(|x|)})$. With s_1 as the advice string, for all strings w, $|w| \leqslant q(|x|)$, $M_{\rm adv}$ correctly decides whether $w \in A$, and thus $M_{\rm sr}$ correctly decides whether $w \in A$ with $M_{\rm adv}$ acting as the oracle. This implies that, for all z and s_2, the A-simulator on input $\langle w, s_1, s_2\rangle$ outputs $M_{\rm adv}(\langle w, s_1\rangle)$, which is the membership of w in A. Thus, the verifier correctly evaluates $B(x, y, z)$. So, $\langle y, s_1\rangle$ is an irrefutable certificate. By symmetry, if $x \notin L$, the No-prover can provide an irrefutable certificate for $x \notin L$. □

A careful examination of the proof of Theorem 4.1 shows that it can be relativized as follows.

Theorem 4.2. *Let A and B be sets such that A is self-reducible with respect to B. If $A \in \mathrm{P}^B/\mathrm{poly}$ then $\mathrm{S}_2^A \subseteq \mathrm{S}_2^B$.*

One direct consequence of Theorem 4.2 is the following corollary.

Corollary 4.3. *If $A \in (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$ and A is Turing self-reducible (or even Turing self-reducible with respect to $\mathrm{NP} \cap \mathrm{coNP}$) then $\mathrm{S}_2^A \subseteq \mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$.*

Proof. Suppose that A is Turing self-reducible with respect to $\mathrm{NP} \cap \mathrm{coNP}$ and that A belongs to $(\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$. Let C be a set in $\mathrm{NP} \cap \mathrm{coNP}$ such that $A \in \mathrm{P}^C/\mathrm{poly}$. Since A is Turing self-re-

ducible with respect to NP ∩ coNP, there is a set D in NP ∩ coNP such that A is Turing self-reducible with respect to D. Select such a D. Let $B = \{0x \mid x \in C\} \cup \{1y \mid y \in D\}$, i.e., let B be the join of C and D. Note that $B \in \mathrm{NP} \cap \mathrm{coNP}$, $A \in \mathrm{P}^B/\mathrm{poly}$, and A is Turing self-reducible with respect to B. By Theorem 4.2, this implies that $\mathrm{S}_2^A \subseteq \mathrm{S}_2^B$. Since $B \in \mathrm{NP} \cap \mathrm{coNP}$, we have $\mathrm{S}_2^A \subseteq \mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$. □

Next, we (as Theorem 4.6) strengthen Corollary 4.3. A statement analogous to Theorem 4.1 for $\mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$ would be that, for every Turing self-reducible set A in $(\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$, A is low for $\mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$, i.e., $\mathrm{S}_2^{\mathrm{NP}^A\cap\mathrm{coNP}^A} = \mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$. This statement would say that, for every NPTM M that is strong with respect to A, if we let $B = L(M^A)$ then $\mathrm{S}_2^B \subseteq \mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$. There are difficulties in proving such a theorem. It seems we need to construct a verifier V running in NP ∩ coNP that can simulate the original verifier V' running in polynomial time with access to $L(M^A)$. To resolve queries to $L(M^A)$ by V', V can simulate M. But to do that V has to answer queries to A. We can use the "dynamic contest" idea as in proof of Theorem 4.1. Each prover can provide a circuit of appropriate size. Then the verifier can first determine the correct circuit using the self-reducing algorithm for A and use it to answer the queries. There are difficulties constructing such a verifier (that runs in NP ∩ coNP). From the definition of $\mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$, note that the verifier must run in NP ∩ coNP for any pair of circuits it is run with respect to. This is not a problem when at least one such circuit is a correct circuit. But the case where both circuits are wrong causes trouble. (This would never really "happen" when working with smart, honest provers, since the prover on the correct "side" would not be so dumb or dishonest as to give a wrong circuit, but nonetheless this is a real problem that potentially blocks membership in NP ∩ coNP.) Note that in this case, verifier V might give wrong answers for queries to A made by the machine M. This amounts to running M with an oracle other than A. If we require only that M is strong with respect to A, M may cease to behave as an NP ∩ coNP machine (i.e., it may have both accepting and rejecting paths on the same input). Thus the verifier V may not be an NP ∩ coNP machine. Although the honest prover has an irrefutable proof, technically speaking V is not an NP ∩ coNP machine. We do not know how to overcome this hurdle. We commend it to the reader as an interesting open issue (and, at the suggestion of a referee, we also suggest that the reader ponder how the classes NP ∩ coNP/poly and NP/poly ∩ coNP/poly relate and potentially differ, see for example the papers [15,18,24]). Here we prove a weaker version, where M is required to be strong with respect to every oracle. To prove the theorem, the following lemma is useful.

Lemma 4.4. *Let A be Turing self-reducible and $A \in \mathrm{P}/\mathrm{poly}$. For each set B, if $B \leq_T^{rs} A$ then $\mathrm{S}_2^B \subseteq \mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$.*

Proof. Let $B \leq_T^{rs} A$ via a robustly strong NPTM M_B. Define the machines M_{adv} and M_{sr}, a "strong advice" function S, a polynomial r, and our A-simulator as in the proof of Theorem 4.1. For each pair of strings s_1 and s_2, define $L_{s_1,s_2} = \{w \mid \text{the } A\text{-simulator accepts} \langle w, s_1, s_2\rangle\}$.

Let $L \in \mathrm{S}_2^B$ via a 3-argument predicate R computable in P^B, and with p being the polynomial setting the length of the proofs. We construct a verifier M, a strong (with respect to the empty set) NPTM machine, to show that $L \in \mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$. M takes as input a string x and certificates $Y = \langle y, s_1\rangle$ and $Z = \langle z, s_2\rangle$ (from the Yes-prover and No-prover, respectively). M simulates R (which is in P^B) on the input $\langle x, y, z\rangle$. During the simulation, suppose a query b is made to the oracle B. To answer the query, M simulates M_B (the strong NPTM that computes B given oracle A) on b. Since M_B is nondeterministic, M branches off into many paths, each path simulating one path of M_B.

Consider any one path. There may be oracle queries. To answer a query a, M uses the A-simulator with input $\langle a, s_1, s_2\rangle$. (Recall that the A-simulator is a deterministic polynomial-time algorithm.) Of these paths some will end with "?" and others with a decisive answer. Observe that all the paths are using the same language L_{s_1,s_2} to answer the oracle queries made by M_B. Since M_B is robustly strong, irrespective of whether L_{s_1,s_2} agrees with A or not, there will be at least one decisive path. Furthermore, although the answer to the query b reached by these decisive paths may not agree with the set B, they all have the same answer. So we let the "?" paths halt and let the decisive paths continue simulating R. Finally, all the paths that finished simulating R (without halting with a "?") will agree on whether or not to accept $\langle x, y, z\rangle$. Again, there will be at least one path that did not halt with a "?" and all such paths have the same outcome, though the outcome may not be the same as $R(x, y, z)$. So, M is a strong (with respect to empty set) NPTM. Thus, by Proposition 3.5, $L(M) \in \mathrm{NP} \cap \mathrm{coNP}$.

As to the correctness of M, note that, for all n, for all strings w such that $|w| \leqslant n$, if either $s_1 = S(1^n)$ or $s_2 = S(1^n)$ then it holds that $w \in A \iff \langle w, s_1, s_2\rangle \in L_{s_1,s_2}$. Suppose $x \in L$. Let q be a polynomial in $|x|$ that bounds the length of queries to A asked by M_B. There exists a y_0 such that, for all z with $|z| = p(|x|)$, it holds that $R(x, y_0, z) = 1$. Now the Yes-prover has an irrefutable proof $Y = \langle y_0, S(1^{q(|x|)})\rangle$. Then, irrespective of the s_2 given by the No-prover, L_{s_1,s_2} will agree with A on all strings of length up to q. So, M will answer all queries to B made by R correctly. Thus, for all Z such that $Z = \langle z, s_2\rangle$, $|z| = p(|x|)$, and $|s_2| \leqslant r(q(|x|))$, it holds that $\langle x, Y, Z\rangle \in L(M)$. By symmetry, if $x \notin L$, then there is some Z such that, for all Y such that $Y = \langle y, s_1\rangle$ with $|y| = p(|x|)$ and $|s_1| \leqslant r(q(|x|))$, it holds that $\langle x, Y, Z\rangle \in \overline{L(M)}$. □

Lemma 4.5. *Let A and D be sets such that A is Turing self-reducible and $A \in \mathrm{P}^D/\mathrm{poly}$. For each set B, if $B \leq_T^{rs} A$ then $\mathrm{S}_2^B \subseteq \mathrm{S}_2[\mathrm{NP}^D \cap \mathrm{coNP}^D]$.*

Proof. We apply the proof of Lemma 4.4. This time the A-simulator runs with oracle D. □

Theorem 4.6. *Let A be a Turing self-reducible (or even self-reducible with respect to $\mathrm{NP} \cap \mathrm{coNP}$) set in $(\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$. For each set B, if $B \leq_T^{rs} A$ then $\mathrm{S}_2^B \subseteq \mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$.*

Proof. The hypothesis of the theorem implies that for some $D \in \mathrm{NP} \cap \mathrm{coNP}$ we have $A \in \mathrm{P}^D/\mathrm{poly}$. By Lemma 4.5, $\mathrm{S}_2^B \subseteq \mathrm{S}_2[\mathrm{NP}^D \cap \mathrm{coNP}^D]$. Since every set in $\mathrm{NP} \cap \mathrm{coNP}$ is low for NP (and so low for coNP), we have $\mathrm{S}_2^B \subseteq \mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$. □

We end this section with some lowness transference lemmas. We show that the low sets of S_2 are also low for $\mathrm{ZPP}^{\mathrm{NP}}$ and $\mathrm{NP}^{\mathrm{NP}}$. To accomplish this, we need a relativized version of Cai's result $\mathrm{S}_2 \subseteq \mathrm{ZPP}^{\mathrm{NP}}$ [13].

Fact 4.7. *For each A, $\mathrm{S}_2^A \subseteq \mathrm{ZPP}^{\mathrm{NP}^A}$.*

The above fact holds via adapting the proof of Cai into a relativized setting. We do not include here a relativized replay of Cai's proof, but rather we explain the only slightly tricky issue in the adaptation. Cai's proof of $\mathrm{S}_2 \subseteq \mathrm{ZPP}^{\mathrm{NP}}$ in [13] goes as follows: Let L be an arbitrary set in S_2. There exist a polynomial p and a polynomial-time predicate P that witness the membership of L in S_2. Cai then designs a ZPP procedure for L that uses SAT as the oracle, where the SAT oracle is used just to check the existence of strings that satisfy (or fail to satisfy) P and to enumerate polynomially many

such strings. Thus, in the case in which the predicate belongs to P^A for some oracle A, the SAT oracle can be replaced by an arbitrary $\leq_m^p$-complete language for NP^A. This implies that $S_2^A \subseteq ZPP^{NP^A}$.

A corollary to Fact 4.7 is the following. (Recall that NP ∩ coNP is low for NP.)

Corollary 4.8. $S_2^{NP\cap coNP} \subseteq ZPP^{NP}$.

We now state our lowness transference lemmas.

Lemma 4.9. *If A is low for S_2 then A is low for ZPP^{NP} and A is low for NP^{NP}.*

Lemma 4.9 is implied by the following stronger transference lemma.

Lemma 4.10. *If $S_2^A \subseteq S_2^{NP\cap coNP}$ then A is low for ZPP^{NP} and A is low for NP^{NP}.*

Proof. Let $S_2^A \subseteq S_2^{NP\cap coNP}$. Then,

$$ZPP^{NP^A} \subseteq ZPP^{S_2^A} \subseteq ZPP^{S_2^{NP\cap coNP}} \subseteq ZPP^{ZPP^{NP}} \subseteq ZPP^{NP}.$$

The first inclusion holds because for each A, $NP^A \subseteq S_2^A$. The second inclusion holds due to the hypothesis of the lemma. The third inclusion is due to Corollary 4.8. The last inclusion is well known (the result $ZPP^{ZPP} = ZPP$ [54] relativizes, see also [33]). So A is low for ZPP^{NP}. Lowness for NP^{NP} follows from this, since Köbler and Watanabe [33] have noted that ZPP^{NP}-lowness implies NP^{NP}-lowness (this holds since—using the standard definition of the ∃ operator and the fact that $\exists \cdot NP^{NP^A} = \exists \cdot \exists \cdot coNP^A = \exists \cdot coNP^A = NP^{NP^A}$—if we assume that A is low for ZPP^{NP} then we obtain $NP^{NP^A} = \exists \cdot coNP^A \subseteq \exists \cdot P^{NP^A} \subseteq \exists \cdot ZPP^{NP^A} = \exists \cdot ZPP^{NP} \subseteq \exists \cdot NP^{NP} = NP^{NP}$). □

5. Applications of S_2 lowness theorems

In this section, we use our lowness theorems proven in Section 4, and some observations we make regarding interactive proof systems, to obtain collapse results for many complexity classes. The following proposition and theorems are useful in proving those results.

Proposition 5.1. $NP \subseteq S_2 \subseteq S_2^{NP\cap coNP} \subseteq \Sigma_2^p \cap \Pi_2^p \subseteq \Sigma_2^p \cup \Pi_2^p \subseteq S_2^{NP} \subseteq PH \subseteq PSPACE$.

We mention in passing that Buhrman and Fortnow (personal communication, 2001) have constructed an oracle separating $\Sigma_2^p \cap \Pi_2^p$ from S_2.

Theorem 5.2. *Let $\mathcal{C}$ be any complexity class that has a self-reducible Turing-complete set. Then $\mathcal{C} \subseteq$ P/poly implies $S_2^{\mathcal{C}} = S_2$. Furthermore, if $S_2 \subseteq \mathcal{C}$, then $\mathcal{C} \subseteq$ P/poly implies $\mathcal{C} = S_2$.*

Proof. Let $B \in \mathcal{C}$ be a self-reducible Turing-complete set for $\mathcal{C}$. Suppose $\mathcal{C} \subseteq$ P/poly. Then $B \in$ P/poly. Applying Theorem 4.1 to B, we get $S_2^B = S_2$. Since B is Turing-complete for $\mathcal{C}$, $S_2^{\mathcal{C}} = S_2^B$. Thus, $S_2^{\mathcal{C}} = S_2$. The second part of the theorem is immediate. □

Theorem 5.3. *Let $\mathcal{C}$ be any complexity class that has a self-reducible Turing-complete set. Then $\mathcal{C} \subseteq$ (NP ∩ coNP)/poly implies $S_2^{\mathcal{C}} \subseteq S_2^{NP\cap coNP}$.*

5.1. Collapse consequences of NP having small circuits

Karp and Lipton [29] showed that if NP $\subseteq$ P/poly then PH $= \Sigma_2^p \cap \Pi_2^p$. Köbler and Watanabe [33] strengthened their result to show that if NP $\subseteq$ (NP $\cap$ coNP)/poly then PH $=$ ZPP$^{\mathrm{NP}}$. The following theorem shows further strengthenings of this result.

Theorem 5.4.

(1) (Hopcroft and Sengupta, see [13]) NP $\subseteq$ P/poly $\Longrightarrow$ PH $= \mathrm{S}_2$.
(2) NP $\subseteq$ (NP $\cap$ coNP)/poly $\Longrightarrow$ PH $= \mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$.

Proof. As to Part 1, we derive the Hopcroft–Sengupta collapse as a consequence of our lowness result. SAT is a self-reducible many-one complete set for NP. Suppose NP $\subseteq$ P/poly. By Theorem 5.2, $\mathrm{S}_2^{\mathrm{NP}} = \mathrm{S}_2$. So by Proposition 5.1, we have $\mathrm{S}_2 = \Sigma_2^p = \Pi_2^p$. And, of course, if $\Sigma_2^p = \Pi_2^p$ then PH $= \Sigma_2^p$.

To prove Part 2, suppose NP $\subseteq$ (NP $\cap$ coNP)/poly. Then, SAT $\in$ (NP $\cap$ coNP)/poly. Thus we have,

$$\Pi_2^p \subseteq \mathrm{S}_2^{\mathrm{NP}} \subseteq \mathrm{S}_2^{\mathrm{SAT}} \subseteq \mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}} \subseteq \Sigma_2^p.$$

The first and last inclusions are from Proposition 5.1. To get the third inclusion, observe that SAT is Turing self-reducible and apply Corollary 4.3. From the above chain of inclusions, we have $\mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}} = \Sigma_2^p = \Pi_2^p$. So PH $= \Sigma_2^p = \mathrm{S}_2^{\mathrm{NP}\cap\mathrm{coNP}}$. □

One might wish to say that "Part 2 of Theorem 5.4 is Part 1 of Theorem 5.4 relativized to NP $\cap$ coNP." This is not a rigorously correct claim, since NP $\cap$ coNP currently is not known to have many-one complete sets (side comment: It is known that NP $\cap$ coNP has many-one complete sets if and only if it has Turing complete sets [19,21]). However, it is intuitively correct, and one can give a proof of Part 2 of Theorem 5.4 along these lines; indeed, that is how the authors first saw this result. To prove the result this way, one would note that Part 1 of Theorem 5.4 relativizes (i.e., one would note the $A = B$ special case of Theorem 5.6), and then would easily sidestep the lack of many-one complete sets for NP $\cap$ coNP via a "set-by-set" argument (just as in [22, Corollary 6], which does exactly this for the Karp–Lipton theorem). However, as discussed in far more detail in Section 2, in this paper we seek to prove not merely collapse results, which after all are probably statements of the form "false implies false," but rather we seek to establish, right now and unconditionally, classes of sets that are simple, i.e., that have lowness properties. Collapse results follow immediately from lowness results (e.g., see our proof above of Part 2 of Theorem 5.4). However, even beyond that, we feel that lowness/self-reducibility-connection results more fully capture and clarify the behavior that yields the collapse results; they are the iceberg of which the conditional-collapse results are the flashy tip.

Theorem 5.4 has consequences in the study of reducing solutions of NP functions. "NP has unique solutions" will be used here to mean that there exists a single-valued, nondeterministic polynomial-time computable (partial) function f such that, for each boolean formula $F \in$ SAT, $f(F)$ outputs a satisfying assignment of F. Informally put, to ask whether NP has unique solutions is to ask whether even a nondeterministic polynomial-time function can cull down to exactly one the

collection of satisfying assignments of satisfiable formulas. It is known that NP has unique solutions if and only if every multivalued NP function has a single-valued refinement (see [24,44] for full details on how these notions are formalized). Hemaspaandra et al. [24] prove that if NP has unique solutions then the polynomial hierarchy collapses to $\mathrm{ZPP}^{\mathrm{NP}}$. Their actual proof involves showing that if NP has unique solutions then $\mathrm{NP} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$. Thus, in light of Part 2 of Theorem 5.4, one has the following stronger collapse consequence from the assumption that NP has unique solutions.

Theorem 5.5. *If* NP *has unique solutions then* $\mathrm{PH} = \mathrm{S}_2^{\mathrm{NP} \cap \mathrm{coNP}}$.

One can for the same reason strengthen to a $\mathrm{PH} = \mathrm{S}_2^{\mathrm{NP} \cap \mathrm{coNP}}$ conclusion a closely related theorem, namely, the result of Buhrman et al. [12] that if, in a certain model of access to partial functions, the solution function of some "universal relation [2]" is computable in the class "$\mathrm{FP}^{\mathrm{NPSV[1]}}$," then $\mathrm{PH} = \mathrm{NP}^{\mathrm{NP}}$.

We now turn to the following result, which shows that Part 1 of Theorem 5.4 can be relativized in a particularly flexible fashion (the natural relativization would simply be the $A = B$ case).

Theorem 5.6. *For any two sets A and B, if* $\mathrm{NP}^A \subseteq \mathrm{P}^B/\mathrm{poly}$ *and* $A \in \mathrm{P}^B$ *then* $\mathrm{NP}^{\mathrm{NP}^A} \subseteq \mathrm{S}_2^B$.

Proof. We first define a language $\mathrm{Univ}_{\mathrm{NP}}^A$. The language $\mathrm{Univ}_{\mathrm{NP}}^A$ consists of all tuples $\langle M, x, y, 1^m, 1^l, 1^{2k} \rangle$ that satisfy (i) M is a deterministic oracle Turing machine, (ii) $|y| + k = l$, and (iii) there is a string w of length l such that y is a prefix of w and M^A accepts $x\#w$ within m steps. In the above construction, the string 1^{2k} is used as padding to help us decrease lengths in such a way as to satisfy the definition of self-reducibility. It is easy to see that $\mathrm{Univ}_{\mathrm{NP}}^A$ is many-one complete for NP^A. Using the given assumption that $A \in \mathrm{P}^B$, we next show that $\mathrm{Univ}_{\mathrm{NP}}^A$ is self-reducible with respect to B: Given input $\langle M, x, y, 1^m, 1^l, 1^{2k} \rangle$, we first verify that condition (ii) is met. Next, if $|y| = l$, we run M on $x\#y$ for m steps, with A as the oracle and accept (reject) if M accepts (rejects). If $|y| < l$, we use the self-reducibility property of $\mathrm{Univ}_{\mathrm{NP}}^A$: $\langle M, x, y, 1^m, 1^l, 1^{2k} \rangle \in \mathrm{Univ}_{\mathrm{NP}}^A$ if and only if $\langle M, x, y0, 1^m, 1^l, 1^{2(k-1)} \rangle \in \mathrm{Univ}_{\mathrm{NP}}^A$ or $\langle M, x, y1, 1^m, 1^l, 1^{2(k-1)} \rangle \in \mathrm{Univ}_{\mathrm{NP}}^A$. Since we added a bit to y but shrunk the final argument by two bits, both the query strings will be shorter than the input string (we are here assuming that the pairing function is such that it enforces this—and we allow the pairing function to be nonsurjective in order to facilitate such enforcement). Thus $\mathrm{Univ}_{\mathrm{NP}}^A$ is self-reducible with respect to A. Since we assumed that $A \in \mathrm{P}^B$, $\mathrm{Univ}_{\mathrm{NP}}^A$ is self-reducible with respect to B. We now use Theorem 4.2 to get $\mathrm{S}_2^{\mathrm{NP}^A} \subseteq \mathrm{S}_2^B$. Since $\mathrm{NP}^{\mathrm{NP}^A} \subseteq \mathrm{S}_2^{\mathrm{NP}^A}$, we have $\mathrm{NP}^{\mathrm{NP}^A} \subseteq \mathrm{S}_2^B$. □

We now immediately apply Theorem 5.6 to improve Yap's theorem. Recall, as mentioned in the introduction, that Yap's theorem [53] is $\mathrm{NP} \subseteq \mathrm{coNP}/\mathrm{poly} \Longrightarrow \mathrm{PH} = \Sigma_3^p$ and that the best previously known strengthening of it is the result of Köbler and Watanabe [33] that $\mathrm{NP} \subseteq \mathrm{coNP}/\mathrm{poly} \Longrightarrow \mathrm{PH} = \mathrm{ZPP}^{\Sigma_3^p}$. We will further strengthen Yap's theorem by proving $\mathrm{NP} \subseteq \mathrm{coNP}/\mathrm{poly} \Longrightarrow \mathrm{PH} = \mathrm{S}_2^{\mathrm{NP}}$. This is a strengthening (in the same standard, informal sense that the strengthenings mentioned above are strengthenings) since by Fact 4.7 it holds that $\mathrm{S}_2^{\mathrm{NP}} \subseteq \mathrm{ZPP}^{\Sigma_2^p}$. Both Yap and Köbler–Watanabe proved their theorems in the more general forms, respectively, "$\Sigma_k^p \subseteq \Pi_k^p/\mathrm{poly} \Longrightarrow \mathrm{PH} = \Sigma_{k+2}^p$, for each $k \geqslant 1$" and "$\Sigma_k^p \subseteq \Pi_k^p/\mathrm{poly} \Longrightarrow \mathrm{PH} = \mathrm{ZPP}^{\Sigma_{k+1}^p}$, for each $k \geqslant 1$," and we also will prove our result in a form broad enough to apply to all levels of the polynomial hierarchy,

namely, we will prove that for each $k \geqslant 1$ it holds that $\Sigma_k^p \subseteq \Pi_k^p/\text{poly} \Longrightarrow \text{PH} = \text{S}_2^{\Sigma_k^p}$. This is a strengthening of Yap and Köbler–Watanabe, since by Fact 4.7 it holds that $\text{S}_2^{\Sigma_k^p} \subseteq \text{ZPP}^{\Sigma_{k+1}^p}$.

Corollary 5.7. *For each* $k \geqslant 1$, $\Sigma_k^p \subseteq \Pi_k^p/\text{poly} \Longrightarrow \text{PH} \subseteq \text{S}_2^{\Sigma_k^p}$. *In particular*, $\text{NP} \subseteq \text{coNP/poly} \Longrightarrow \text{PH} \subseteq \text{S}_2^{\text{NP}}$.

Proof. Assume $\Pi_k^p \subseteq \Sigma_k^p/\text{poly}$, which is equivalent to our hypothesis. This easily implies (as observed in the proof of [33, Corollary 4.4]) that $\Sigma_{k+1}^p \subseteq \Sigma_k^p/\text{poly}$, and thus certainly that $\Sigma_{k+1}^p \subseteq \text{P}^{\Sigma_k^p}/\text{poly}$. Now, invoking Theorem 5.6 with A and B both chosen to be some $\leq_m^p$-complete set for Σ_k^p, we have that $\Sigma_{k+2}^p \subseteq \text{S}_2^{\Sigma_k^p}$. Since $\text{S}_2^{\Sigma_k^p} \subseteq \Pi_{k+2}^p$ (see Fact 4.7), this implies $\text{PH} = \text{S}_2^{\Sigma_k^p}$. □

Note that the final sentence of the statement of Corollary 5.7 strengthens all known results that are based on Yap's theorem. We limit ourselves to two brief examples. First, the core tool of Chang's [17] very interesting work on approximations and bounded queries is a lemma of Wagner [52] (cited in Chang's [17] paper as Theorem 17 there). The tool is used by Chang's approximation paper to conclude, under a broad variety of assumptions, that the polynomial hierarchy collapses to Σ_3^p. However, the tool's proof itself, as one can see clearly from Chang's sketch of its proof, internally obtains that (from its assumptions) $\text{NP} \subseteq \text{coNP/poly}$, and from there invokes Yap's theorem to conclude $\text{PH} = \Sigma_3^p$. Thus, the final sentence of Corollary 5.7 in fact strengthens, to S_2^{NP}, the collapses that that tool broadly yields.

Our second example regards the lovely paper "Improving Known Solutions is Hard," by Ranjan et al. [39]. That paper proves that "if $\text{PH} \neq \Sigma_3^p$, [then] polynomial-time transducers cannot compute optimal solutions for many problems, even given $n^{1-\epsilon}$ *nontrivial* solutions, for any $\epsilon > 0$; ... and [they show that their] results hold even in the presence of randomness" (see their paper for definitions of their model, and for their many precisely stated theorems). Their paper repeatedly obtains results containing in their statements "if $\text{PH} \neq \Sigma_3^p$" and (though they actually simply write "unless PH collapses," what they actually show is the following) "unless $\text{PH} = \Sigma_3^p$." However, as one can see clearly from their proofs, these statements are coming in due to Yap's theorem, and thus the theorems can, due to the final sentence of the statement of Corollary 5.7, be, respectively, improved to "if $\text{PH} \neq \text{S}_2^{\text{NP}}$" and "unless $\text{PH} = \text{S}_2^{\text{NP}}$." Thus, their sufficient condition for ensuring that improving known solution is hard can be weakened; that is, their results, when improved in light of the results of our paper, provide even stronger evidence that improving known solutions is hard.

Part 2 of Theorem 5.4 improved the collapse consequences that follow from $\text{NP} \subseteq (\text{NP} \cap \text{coNP})/\text{poly}$. In Section 5.2 we will similarly improve such collapse consequences for the cases of various other classes, such as UP and FewP. In each case, the conclusions involve S_2-related classes.

We will not extensively discuss what happens if large classes such as Mod-based ones are contained in P/poly or $(\text{NP} \cap \text{coNP})/\text{poly}$, though we will provide some improvements. The reason we will not discuss this extensively is that the assumption that P/poly or $(\text{NP} \cap \text{coNP})/\text{poly}$ contains such powerful classes (classes that have suitable interactive proof systems, and due to Toda's theorem, in the presence of the BP operator subsume the polynomial hierarchy [46,47]) is so sweeping as to immediately imply collapses that are even deeper than S_2-related collapses. In particular, it is known that the following holds.

Theorem 5.8 (see [33, 36])**.**

(1) *If* $\mathrm{PP} \subseteq \mathrm{P/poly}$, *then* $\mathrm{P}^{\#\mathrm{P}} = \mathrm{MA}$.
(2) *If* $\mathrm{PSPACE} \subseteq \mathrm{P/poly}$, *then* $\mathrm{PSPACE} = \mathrm{MA}$.
(3) *If* $\mathrm{Mod}_k\mathrm{P} \subseteq \mathrm{P/poly}$, *then* $\mathrm{Mod}_k\mathrm{P} \subseteq \mathrm{MA}$.

Theorem 5.8 is proved via the following line of argument. We start with a class $\mathcal{C}$ with a many-one complete language $L_{\mathcal{C}}$ that has an interactive proof system in which the power of the prover lies in $\mathrm{FP}^{\mathcal{C}}$. We then argue that, $\mathcal{C} \subseteq \mathrm{P/poly}$ implies $\mathcal{C} \subseteq \mathrm{MA}$. As the classes PP, PSPACE, and $\mathrm{Mod}_k\mathrm{P}$ all have the required many-one complete languages, we obtain Theorem 5.8.

We now (as Theorems 5.9 and 5.10) generalize the above line of argument, and then (as Theorem 5.13) via our generalization obtain strong new collapse results regarding $\mathrm{Mod}_k\mathrm{P}$ classes.

Theorem 5.9. *Let $\mathcal{C}$ be a class closed under polynomial-time many-one reductions and that has a many-one complete language $L_{\mathcal{C}}$ such that both $L_{\mathcal{C}}$ and its complement $\overline{L_{\mathcal{C}}}$ have interactive proof systems in which the power of the honest prover lies in $\mathrm{FP}^{\mathcal{C}}$. Then the following hold.*

(1) $\mathcal{C} \subseteq \mathrm{P/poly} \Longrightarrow \mathrm{BPP}^{\mathcal{C}} \subseteq \mathrm{MA}$.
(2) $\mathcal{C} \subseteq \mathrm{P/poly} \Longrightarrow \mathrm{NP}^{\mathcal{C}} \subseteq \mathrm{MA}$.

Proof. Our proof is in spirit related to the proof of [36, Corollary 8]. We first prove Part 1. Note that the statement clearly holds for $\mathcal{C} = \{\Sigma^*\}$. So, we will assume that $\mathcal{C}$ contains a set that is not Σ^*. Since the only class for which Σ^* is many-one complete is the class $\{\Sigma^*\}$, this ensures that the many-one complete set $L_{\mathcal{C}}$ here is not Σ^*. Suppose $\mathcal{C} \subseteq \mathrm{P/poly}$. Let $L_0 = \{1^k 0 y \mid k \geqslant 0 \wedge y \in L_{\mathcal{C}}\}$. It is easy to see that L_0 is polynomial-time many-one equivalent to $L_{\mathcal{C}}$. Since $L_{\mathcal{C}}$ is polynomial-time many-one complete for $\mathcal{C}$ and $\mathcal{C}$ is closed under polynomial-time many-one reductions, it holds that L_0 is polynomial-time many-one complete for $\mathcal{C}$. Since $L_{\mathcal{C}}$ and $\overline{L_{\mathcal{C}}}$ have interactive proof systems, both L_0 and $\overline{L_0}$ have interactive proof systems.

Since L_0 is many-one complete for $\mathcal{C}$, it suffices to show that $\mathrm{BPP}^{L_0} \subseteq \mathrm{MA}$. Since $L_0 \in \mathcal{C}$ and $\mathcal{C} \subseteq \mathrm{P/poly}$, let M_{adv} be a polynomial-time machine that accepts L_0 given appropriate advice. Let $A \in \mathrm{BPP}^{L_0}$ via a bounded-error probabilistic polynomial-time oracle Turing machine M such that $A = L(M^{L_0})$. It is well known that for any given polynomial $t(n)$ the error probability of a BPP-machine can be made amplified to being at most $2^{-t(n)}$, namely, by taking the majority vote of polynomially many independent runs of the machine (see, e.g. [42]). Using the fact we can assume that for all x the error probability of M on input x is at most $2^{-t(|x|)}$, where t is a polynomial that will be defined later. On an input of length n, the machine M can ask queries of length up to some l (which is polynomial in n). Since the exponential reduction in the error probability of M is achieved by independent simulations of M, this polynomial l can be determined independently of the choice of t (namely, whatever l-bound holds for one simulation of M will also hold for our polynomial number of simulations).

By our assumption, there are interactive proof systems for both L_0 and $\overline{L_0}$ in which the power of the honest prover lies in FP^{L_0}. Since the computational power of honest provers exists in a deterministic complexity class, we can assume that the honest provers are deterministic, and thus, all the provers are deterministic. As we did for the machine M, we can assume that the error proba-

bilities that characterize the soundness and correctness of the interactive proof systems have been exponentially reduced. Indeed, we can assume that the same function $2^{-t(n)}$ is used as the bound. That is, for each $x \in L_0$ (respectively, for each $x \notin L_0$), the probability that the system for L_0 (respectively, the system for $\overline{L_0}$) rejects given an honest prover is at most $2^{-t(|x|)}$, and for each $x \notin L_0$ (respectively, for each $x \in L_0$), the probability that the system for L_0 (respectively, the system for $\overline{L_0}$) accepts given any prover is at most $2^{-t(|x|)}$. Let m' be a polynomial such that, for each $n \geqslant 0$, $m'(n)$ bounds from the above the length of the longest query to the oracle L_0 asked by the honest prover, when running the interactive proof systems for L_0 and $\overline{L_0}$ on strings of length up to $l(n)$. Again, this polynomial m' is independent of the polynomial t. Define $m(n) = l(n) + m'(n)$. Then for all n it holds that $m(n) \geqslant \max(l(n), m'(n))$. Let r be the length of advice needed by M_{adv} on strings of length up to m. Clearly, r is polynomial in n.

We now exhibit a Merlin–Arthur game for A. The protocol for an input x is as follows. Merlin first gives a string s of length $r(|x|)$. (Informally, the hope is that s is a "good" advice string given which M_{adv} can correctly decide membership in L_0 for all inputs of length up to $m(|x|)$. Of course, Merlin may be cheating by giving a "bad" advice string.) Arthur simulates the BPP oracle machine M on input x. Suppose M asks a query q. If q is not of the form $1^k 0 y$, q is trivially a nonmember of L_0, so Arthur returns to the simulation assuming that the oracle answer is in the negative. If q is of the form $1^k 0 y$ but its length is less than $m(|x|)$, then Arthur modifies q by attaching an appropriate number of 1's at the beginning of the string so that q has length exactly $m(|x|)$. The membership of this new string in L_0 is identical to the membership of q in the original form, so Arthur can use this new q instead of the original q. Arthur then runs $M_{\mathrm{adv}}(\langle q, s \rangle)$ and obtains an answer $a \in \{accept, reject\}$ (which may be incorrect since the string s given by Merlin may be a "bad" advice string). Arthur verifies correctness of the answer by simulating the interactive proof system for either L_0 or $\overline{L_0}$ on query q, depending on whether $a = accept$ or $a = reject$, respectively. Since the power of the honest prover lies in FP^{L_0}, Arthur can simulate the prover. He answers queries of the prover to L_0 by simulating M_{adv} on the query with s as the advice. After running the proof system, if he is convinced that a is the correct answer to query q, he continues to simulate the machine M on x. If he is not convinced, he simply halts, and rejects the input x. He performs the above-specified actions whenever M asks a query. Finally, if M runs to completion, Arthur outputs the outcome of M.

Certainly, the game defined in the above runs in polynomial time. Let u be a polynomial bounding the running time of the game. Suppose $x \in A$. Then Merlin can give a good advice string s. Given this advice string, M_{adv} correctly decides the membership in L_0 for all strings of length up to $m(|x|)$. So, for each query q of M on input x, Arthur obtains the correct membership of q in L_0 by simulating M_{adv}. Also, with this advice string s, for each query q of M on input x, the computation of the honest prover can be correctly simulated for both L_0 and $\overline{L_0}$. Since each query of M on input x has length $m(|x|)$, the probability that Arthur is not convinced of the correctness of the answer it obtains for a query q of M on x is at most $2^{-t(m(|x|))}$. The machine M on x with oracle L_0 rejects with probability at most $2^{-t(|x|)}$. The number of queries that Arthur makes on input x is certainly bounded by the running time, so it is at most $u(|x|)$. So, the probability that the simulation of M deviates from the correct one (via obtaining an incorrect oracle answer) is at most $u(|x|)2^{-t(m(|x|))}$. Thus, the probability that Arthur rejects x given this correct advice string s is at most

$$2^{-t(|x|)} + \left(1 - 2^{-t(|x|)}\right) u(|x|) \left(2^{-t(m(|x|))}\right).$$

This quantity is at most

$$2^{-t(|x|)} + u(|x|)\left(2^{-t(m(|x|))}\right).$$

Choose the polynomial t so that for all $n \geqslant 0$ it holds that $t(n) \geqslant 3$ and $t(m(n)) \geqslant \lceil \log_2 u(n) \rceil + 3$. Then, the probability in question is at most $\frac{1}{4}$. Hence, the probability that Arthur rejects x given this correct advice string s is at most $\frac{1}{4}$.

On the other hand, suppose $x \notin A$. Let s be an arbitrary string of length r given by Merlin. In this case, M on x accepts with probability at most $2^{-t(|x|)}$ given L_0 as the oracle. Arthur may get an incorrect answer for a query q, since he runs M_{adv} using the (arbitrary) string s as advice. However, when Arthur uses the interactive proof system to verify the answer, the probability that he is convinced that the answer is correct is at most $2^{-t(m(|x|))}$ (simply because the answer is incorrect, and by the properties of the interactive proof system, *any* prover has only a low probability of convincing the verifier that the answer is correct). Hence, by following an analysis similar to the previous case, the probability of Arthur accepting x is at most $\frac{1}{4}$ regardless of the advice string s. Hence, $A \in \mathrm{MA}$.

Part 2 is proved along similar lines, with the following changes. Apart from the advice string s, Merlin also provides a computational path of the NP oracle machine. Then Arthur simulates the NP machine along the given path. □

By examining the proof of Theorem 5.9, we can obtain the following relativized version.

Theorem 5.10. *Let $\mathcal{C}$ be a class closed under polynomial-time many-one reductions and that has a many-one complete language $L_{\mathcal{C}}$ such that both $L_{\mathcal{C}}$ and its complement $\overline{L}_{\mathcal{C}}$ have interactive proof systems in which the power of the honest prover lies in $\mathrm{FP}^{\mathcal{C}}$. Let A be any language. Then the following hold.*

(1) $\mathcal{C} \subseteq \mathrm{P}^A/\mathrm{poly} \Longrightarrow \mathrm{BPP}^{\mathcal{C}} \subseteq \mathrm{MA}^A$.
(2) $\mathcal{C} \subseteq \mathrm{P}^A/\mathrm{poly} \Longrightarrow \mathrm{NP}^{\mathcal{C}} \subseteq \mathrm{MA}^A$.

We can relax the hypotheses of Theorem 5.9, and obtain some weaker conclusions as follows.

Theorem 5.11. *Let $\mathcal{C}$ be a class closed under polynomial-time many-one reductions and that has a many-one complete language $L_{\mathcal{C}}$ having an interactive proof system in which the power of the honest prover lies in $\mathrm{FP}^{\mathcal{C}}$. Then the following hold.*

(1) $\mathcal{C} \subseteq \mathrm{P}/\mathrm{poly} \Longrightarrow \mathrm{BP} \cdot \mathcal{C} \subseteq \mathrm{MA}$.
(2) $\mathcal{C} \subseteq \mathrm{P}/\mathrm{poly} \Longrightarrow \exists \cdot \mathcal{C} \subseteq \mathrm{MA}$.

The class NP has a many-one complete language SAT that has an interactive proof system in which the power of the prover lies in $\mathrm{FP}^{\mathrm{NP}}$ (a honest prover can find a satisfying truth assignment using self-reducibility). Thus, applying Theorem 5.11 to NP, we can obtain the following known result.

Corollary 5.12 ([4]). *If* $\mathrm{NP} \subseteq \mathrm{P}/\mathrm{poly}$ *then* $\mathrm{AM} = \mathrm{MA}$.

For $k \geqslant 2$, we could apply Theorems 5.9 and 5.10 to the class $\mathrm{Mod}_k\mathrm{P}$, as these classes satisfy the required properties (see also the related discussion within the proof of Theorem 5.13). But by

applying these results in concert with a result of Toda and Ogihara, we can obtain even stronger results.

Theorem 5.13. *Fix* $k \geqslant 2$.

(1) *If* $\mathrm{Mod}_k\mathrm{P} \subseteq \mathrm{P/poly}$ *then* $\mathrm{PH}^{\mathrm{Mod}_k\mathrm{P}} = \mathrm{MA}$.
(2) *Let A be a language in* PH. *If* $\mathrm{Mod}_k\mathrm{P} \subseteq \mathrm{P}^A/\mathrm{poly}$ *then* $\mathrm{PH}^{\mathrm{Mod}_k\mathrm{P}} = \mathrm{MA}^A$.
(3) *If* $\mathrm{Mod}_k\mathrm{P} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$ *then* $\mathrm{PH}^{\mathrm{Mod}_k\mathrm{P}} = \mathrm{MA}^{\mathrm{NP}\cap\mathrm{coNP}}$.

Proof. Let us first prove Part 1. Fix $k \geqslant 2$ and suppose $\mathrm{Mod}_k\mathrm{P} \subseteq \mathrm{P/poly}$. $\mathrm{Mod}_k\mathrm{P}$ has the following many-one complete language.

$$\mathrm{SAT}_k = \{\langle \varphi, r\rangle | \text{the number of satisfying truth assignments of } \varphi \text{ is } r \text{ modulo } k\}.$$

Let $\#\mathrm{P}_k$ denote the class of all functions f for which there exists some $g \in \#\mathrm{P}$ such that, for all x, it holds that $f(x) = (g(x) \bmod k)$ [5]. Observe that $\mathrm{P}^{\mathrm{Mod}_k\mathrm{P}} = \mathrm{P}^{\#\mathrm{P}_k}$. Babai and Fortnow [5] show that $\#\mathrm{P}_k$ has an interactive proof system in which the power of the honest prover lies in $\mathrm{FP}^{\#\mathrm{P}_k}$. Thus, every language in $\mathrm{P}^{\mathrm{Mod}_k\mathrm{P}}$ has a proof system in which the power of the honest prover lies in $\mathrm{FP}^{\mathrm{Mod}_k\mathrm{P}}$. In particular, the language SAT_k and its complement have such proof systems. Thus $\mathrm{Mod}_k\mathrm{P}$ satisfies all the requirements of Theorem 5.9. Applying Theorem 5.9 to $\mathrm{Mod}_k\mathrm{P}$, it follows that $\mathrm{BPP}^{\mathrm{Mod}_k\mathrm{P}} \subseteq \mathrm{MA}$. By a result of Toda and Ogihara [47], it unconditionally holds that $\mathrm{PH} \subseteq \mathrm{BPP}^{\mathrm{Mod}_k\mathrm{P}}$. It follows that $\mathrm{PH} = \mathrm{MA}$. $\mathrm{BPP}^{\mathrm{Mod}_k\mathrm{P}} \subseteq \mathrm{MA}$ trivially implies that $\mathrm{Mod}_k\mathrm{P} \subseteq \mathrm{MA}$. We conclude that

$$\mathrm{PH}^{\mathrm{Mod}_k\mathrm{P}} \subseteq \mathrm{PH}^{\mathrm{MA}} \subseteq \mathrm{PH} = \mathrm{MA}.$$

The second inclusion uses the fact that $\mathrm{MA} \subseteq \Sigma_2^p$ (unconditionally). The second part of the theorem can be proved by using Theorem 5.10 instead of Theorem 5.9 in the above proof. For Part 3, apply Part 2 for each language $A \in \mathrm{NP} \cap \mathrm{coNP}$ individually. □

Finally, let us briefly discuss what holds for PP and PSPACE. Note that the first part of Theorem 5.8 can be enhanced from $\mathrm{P}^{\#\mathrm{P}}$ to $\mathrm{PH}^{\#\mathrm{P}}$ as follows. This enhanced theorem will be useful later when we study the class $\mathrm{C}_=\mathrm{P}$.

Corollary 5.14. *If* $\mathrm{PP} \subseteq \mathrm{P/poly}$, *then* $\mathrm{PH}^{\#\mathrm{P}} = \mathrm{MA}$.

Proof. From Theorem 5.8, the hypothesis certainly implies $\mathrm{P}^{\#\mathrm{P}} = \mathrm{MA}$. So using this we have $\mathrm{PH}^{\#\mathrm{P}} = \mathrm{PH}^{\mathrm{P}^{\#\mathrm{P}}} = \mathrm{PH}^{\mathrm{MA}}$. But, unconditionally, $\mathrm{PH}^{\mathrm{MA}} \subseteq \mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}}$ (which itself we already know is, under the hypothesis, contained in MA). So the hypothesis indeed implies that $\mathrm{PH}^{\#\mathrm{P}} = \mathrm{MA}$. □

We now may state the following.

Theorem 5.15.

(1) *If* $\mathrm{PP} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$, *then* $\mathrm{PH}^{\#\mathrm{P}} = \mathrm{MA}^{\mathrm{NP}\cap\mathrm{coNP}}$.
(2) *If* $\mathrm{PSPACE} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$, *then* $\mathrm{PSPACE} = \mathrm{MA}^{\mathrm{NP}\cap\mathrm{coNP}}$.

One cannot get these claims from the earlier theorems simply by saying "relativize by NP ∩ coNP." This is due to the problem that was mentioned in the comments made immediately after the

proof of Theorem 5.4. In particular, the problem is that NP ∩ coNP has no known (many-one or, equivalently for that particular class, Turing) complete sets, so saying "relativize by NP ∩ coNP" is not a legal step. However, one can use a "set-by-set" argument (see [22, Corollary 6]). Indeed, since PP has many-one complete sets, we can actually use not a "set-by-set" approach but can fix on a single set that, while potentially not many-one complete for NP ∩ coNP will be good enough for the purposes of our proof. To be explicit about this, let us prove the first part of the above theorem. Assume PP ⊆ (NP ∩ coNP)/poly. Let B be an arbitrary many-one complete set for PP. So there is a set $A \in \mathrm{NP} \cap \mathrm{coNP}$ such that $B \in \{A\}/\mathrm{poly}$. Due to B's many-one completeness, we easily have that $\mathrm{PP} \subseteq \mathrm{P}^A/\mathrm{poly}$. By Theorem 5.10 (either part), which applies since for PP there is a many-one complete language that has an interactive proof system in which the power of the honest prover belongs to $\mathrm{FP}^{\mathrm{PP}}$, we certainly may conclude that $\mathrm{P}^{\mathrm{PP}} \subseteq \mathrm{MA}^{\mathrm{NP}\cap\mathrm{coNP}}$. But, given that $\mathrm{P}^{\mathrm{PP}} \subseteq \mathrm{MA}^{\mathrm{NP}\cap\mathrm{coNP}}$, note that $\mathrm{PH}^{\#\mathrm{P}} \subseteq \mathrm{PH}^{\mathrm{PP}} \subseteq \mathrm{PH}^{\mathrm{MA}^{\mathrm{NP}\cap\mathrm{coNP}}} \subseteq \mathrm{PH} \subseteq \mathrm{P}^{\mathrm{PP}} \subseteq \mathrm{MA}^{\mathrm{NP}\cap\mathrm{coNP}} \subseteq \mathrm{PH}^{\#\mathrm{P}}$. So $\mathrm{PH}^{\#\mathrm{P}} = \mathrm{MA}^{\mathrm{NP}\cap\mathrm{coNP}}$ as desired. Since for PSPACE there is a many-one complete language that has an interactive proof system in which the power of the honest prover belongs to $\mathrm{FP}^{\mathrm{PSPACE}}$, one can similarly conclude the other part of the above theorem.

5.2. Results in the absence of self-reducible many-one complete sets

Theorem 4.1 connects nonuniform containments with uniform collapse consequences for classes having self-reducible many-one complete sets. But there are some complexity classes for which self-reducible many-one complete sets are not known. In this section, we handle some such classes.

We first show that Theorem 4.1 can even be applied to some classes $\mathcal{C}$ that potentially lack many-one complete sets, but that do satisfy $\mathcal{C} \subseteq \mathrm{R}_m^p(\mathcal{C} \cap \text{Turing-self-reducible})$, i.e., classes whose Turing self-reducible sets are known to be, in some sense, "collectively" many-one complete for the class. For example, neither UP nor FewP is known to have many-one complete sets, and indeed each is known to lack many-one complete sets—and even Turing-complete sets—in relativized worlds [20,23]. Nonetheless we have the following claim.

Theorem 5.16. *Let $\mathcal{C}$ be any member of this list:* $\mathrm{UP}, \mathrm{coUP}, \mathrm{FewP}, \mathrm{coFewP}$. *If $\mathcal{C} \subseteq \mathrm{P/poly}$ then $\mathcal{C}$ is low for each of* $\mathrm{S}_2, \mathrm{ZPP}^{\mathrm{NP}}$, *and* $\mathrm{NP}^{\mathrm{NP}}$.

Proof. We prove just the UP case. The FewP case is similar. The coUP and coFewP cases follow from the UP and FewP cases, respectively, since

$$\mathcal{C} \subseteq \mathrm{P/poly} \iff \mathrm{co}\,\mathcal{C} \subseteq \mathrm{P/poly}.$$

Assume $\mathrm{UP} \subseteq \mathrm{P/poly}$. Let B be an arbitrary set in $\mathrm{S}_2^{\mathrm{UP}}$. Let $D \in \mathrm{UP}$ be a set such that $B \in \mathrm{S}_2^D$. For each set $E \in \mathrm{UP}$, there exists a 2-disjunctively self-reducible set $A_E \in \mathrm{UP}$ such that $E \leq_m^p A_E$. (For example, if E is accepted by UP-like machine M which runs in time $n^k + k$, then we can take A_E to be, with the pairing function appropriately chosen, $\{\langle x, u, 0^j\rangle \mid 2(|x|^k + k - |u|) = j$ and there is an accepting path of M on input x having u as a prefix$\}$.) So, since $\mathrm{UP} \subseteq \mathrm{P/poly}$, by Theorem 4.1 we have $\mathrm{S}_2^{A_D} = \mathrm{S}_2$. Since $D \leq_m^p A_D$, $\mathrm{S}_2^D \subseteq \mathrm{S}_2^{A_D} = \mathrm{S}_2$. It follows that $\mathrm{S}_2^{\mathrm{UP}} = \mathrm{S}_2$. By Lemma 4.9 we obtain lowness for $\mathrm{ZPP}^{\mathrm{NP}}$ and $\mathrm{NP}^{\mathrm{NP}}$. □

It is easy to see that (i) for each $E \in \mathrm{UP}$ there is a set A_E that is in UP, is Turing (and even 2-disjunctively) self-reducible, and satisfies $E \leqslant_m^p A_E$ (in fact, this in effect has just been used in the

proof of Theorem 5.16); and (ii) for each $E \in \mathrm{FewP}$ there is a set A_E that is in FewP, is Turing (and even 2-disjunctively) self-reducible, and that satisfies $E \leqslant_m^p A_E$. With these facts in hand, it is easy in light of Corollary 4.3 to obtain the following theorem.

Theorem 5.17.

(1) $\mathrm{UP} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly} \Longrightarrow \mathrm{S}_2^{\mathrm{UP}} \subseteq \mathrm{S}_2^{\mathrm{NP} \cap \mathrm{coNP}}$.
(2) $\mathrm{FewP} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly} \Longrightarrow \mathrm{S}_2^{\mathrm{FewP}} \subseteq \mathrm{S}_2^{\mathrm{NP} \cap \mathrm{coNP}}$.

The final complexity class that we consider is $\mathrm{C}_=\mathrm{P}$ [45,51]. (A set A is in $\mathrm{C}_=\mathrm{P}$ exactly if there is a #P function f and a polynomial-time computable function g such that, for each x, it holds that $x \in A \iff f(x) = g(x)$.) Our goal is to prove a complexity class collapse from the assumption that $\mathrm{C}_=\mathrm{P} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$. However, $\mathrm{C}_=\mathrm{P}$ is not known to have self-reducible many-one complete sets, and so Theorem 4.1 cannot be used directly. We will use the following lemma to gain some leverage against the class $\mathrm{C}_=\mathrm{P}$.

Lemma 5.18. *If* $\mathrm{C}_=\mathrm{P} \subseteq \mathrm{P}/\mathrm{poly}$ *then* $\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{P}/\mathrm{poly}$.

Proof. Suppose $\mathrm{C}_=\mathrm{P} \subseteq \mathrm{P}/\mathrm{poly}$. Since $\mathrm{P}^{\#\mathrm{P}} = \mathrm{P}^{\mathrm{PP}}$ and $\mathrm{NP}^{\mathrm{PP}} = \mathrm{NP}^{\mathrm{C}_=\mathrm{P}}$ (due to Torán [48]), we have $\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{NP}^{\mathrm{C}_=\mathrm{P}}$, so $\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{NP}/\mathrm{poly}$. Since $\mathrm{coNP} \subseteq \mathrm{C}_=\mathrm{P}$ and $\mathrm{co}(\mathrm{P}/\mathrm{poly}) = \mathrm{P}/\mathrm{poly}$, we have $\mathrm{NP}/\mathrm{poly} \subseteq (\mathrm{P}/\mathrm{poly})/\mathrm{poly} = \mathrm{P}/\mathrm{poly}$. Thus, $\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{P}/\mathrm{poly}$. □

Theorem 5.19. *If* $\mathrm{C}_=\mathrm{P} \subseteq \mathrm{P}/\mathrm{poly}$ *then* $\mathrm{PH}^{\#\mathrm{P}} = \mathrm{PH}^{\mathrm{C}_=\mathrm{P}} = \mathrm{S}_2^{\mathrm{C}_=\mathrm{P}} = \mathrm{S}_2 = \mathrm{MA}$.

Proof. By our assumption and Lemma 5.18, $\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{P}/\mathrm{poly}$. This implies $\mathrm{PP} \subseteq \mathrm{P}/\mathrm{poly}$. So by part 1 of Theorem 5.8 we have $\mathrm{PH}^{\#\mathrm{P}} = \mathrm{MA}$. It holds unconditionally that $\mathrm{MA} \subseteq \mathrm{S}_2$, $\mathrm{S}_2^{\mathrm{C}_=\mathrm{P}} \subseteq \mathrm{PH}^{\#\mathrm{P}}$, and $\mathrm{PH}^{\#\mathrm{P}} = \mathrm{PH}^{\mathrm{C}_=\mathrm{P}}$. So we have the desired conclusion. □

The following result shows what holds under the assumption $\mathrm{C}_=\mathrm{P} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$.

Theorem 5.20. *If* $\mathrm{C}_=\mathrm{P} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$ *then* $\mathrm{PH}^{\#\mathrm{P}} = \mathrm{MA}^{\mathrm{NP} \cap \mathrm{coNP}}$.

Proof. Suppose that $\mathrm{C}_=\mathrm{P} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$. Then $\mathrm{coC}_=\mathrm{P} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$. Since $\mathrm{NP}^{\mathrm{PP}} = \mathrm{NP}^{\mathrm{C}_=\mathrm{P}}$ (again due to Torán [48]), $\mathrm{PP} \subseteq \mathrm{NP}^{\mathrm{PP}} \subseteq \mathrm{NP}^{\mathrm{C}_=\mathrm{P}} \subseteq \mathrm{NP}^{(\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}} \subseteq \mathrm{NP}^{\mathrm{NP} \cap \mathrm{coNP}}/\mathrm{poly}$, and so $\mathrm{PP} \subseteq \mathrm{NP}^{\mathrm{NP} \cap \mathrm{coNP}}/\mathrm{poly}$. Since $\mathrm{NP}^{\mathrm{NP} \cap \mathrm{coNP}} = \mathrm{NP}$ and $\mathrm{NP} \subseteq \mathrm{coC}_=\mathrm{P}$, we have $\mathrm{PP} \subseteq \mathrm{NP}/\mathrm{poly} \subseteq \mathrm{coC}_=\mathrm{P}/\mathrm{poly} \subseteq ((\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly})/\mathrm{poly} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$. By part 1 of Theorem 5.15, this implies that $\mathrm{PH}^{\#\mathrm{P}} = \mathrm{MA}^{\mathrm{NP} \cap \mathrm{coNP}}$. □

6. Open questions

We state some open issues. Can Theorem 4.1 be strengthened to the following claim: If A is Turing self-reducible and $A \in (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$, then A is low for S_2? Can one prove that $\mathrm{S}_2^{\mathrm{NP} \cap \mathrm{coNP}} = \mathrm{S}_2$? The latter would imply the former. The questions touch upon an interesting issue, namely, before the work of Cai/Hopcroft–Sengupta, for over a decade $\mathrm{NP} \subseteq \mathrm{P}/\mathrm{poly}$ and $\mathrm{NP} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$ were thought to imply the same strength of collapse: first $\mathrm{PH} = \mathrm{NP}^{\mathrm{NP}}$ [1,22,28,29] and later $\mathrm{PH} = \mathrm{ZPP}^{\mathrm{NP}}$ [33].

Cai's result $S_2 \subseteq ZPP^{NP}$ shows that the Hopcroft–Sengupta consequence from $NP \subseteq P/poly$, namely $PH = S_2$, is actually the strongest currently known collapse that follows from $NP \subseteq P/poly$. The present paper shows that $NP \subseteq (NP \cap coNP)/poly$ implies $PH = S_2^{NP \cap coNP}$. This is the strongest currently known collapse from the assumption $NP \subseteq (NP \cap coNP)/poly$. Thus the best collapses known from $NP \subseteq P/poly$ and $NP \subseteq (NP \cap coNP)/poly$ currently differ.

Cai et al. [15] have given a fine-grained analysis of robustly strong reductions and so have shown strengthenings of previous Karp–Lipton-type results. It would be interesting to see whether a similar "fine structure of reductions" approach could yield strengthening in the present context.

Finally, our lowness results are stated for Turing self-reducibility. However, as mentioned in Section 3, our results even hold for "nice-p-order"-based self-reducibility, a more general notion that allows self-reduction trees to have polynomially long paths (rather than the linearly-long paths of Turing self-reducibility trees). Though a wide array of natural classes have many-one complete sets that have both types of self-reducibility, researchers have also studied even more general notions of self-reducibility that capture some additional complexity classes such as $C_=P$ (see [6]), via allowing very long paths. Such notions even have lowness theorems supporting Σ_2^p-type and ZPP^{NP}-type results (see [6,33]). Can one extend our lowness results to such extremely general self-reducibility notions? It seems that doing so would require a proof approach substantially different from that of Theorem 4.1, since the dynamic contest procedure in the proof of Theorem 4.1 makes the proof crucially depend on the self-reducibility trees having polynomially bounded depth.

Acknowledgments

We thank Edith Hemaspaandra for helpful discussions and comments, the anonymous referees for helpful comments, Vinodchandran Variyam for exchanging papers with us, and Alina Beygelzimer, Mayur Thakur, and Rahul Tripathi for proofreading drafts of this paper.

References

[1] M. Abadi, J. Feigenbaum, J. Kilian, On hiding information from an oracle, Journal of Computer and System Sciences 39 (1989) 21–50.
[2] M. Agrawal, S. Biswas, Universal relations, in: Proceedings of the 7th structure in complexity theory conference, IEEE Computer Society Press, Silver Spring, MD, 1992, pp. 207–220.
[3] V. Arvind, Y. Han, L. Hemachandra, J. Köbler, A. Lozano, M. Mundhenk, M. Ogiwara, U. Schöning, R. Silvestri, T. Thierauf, Reductions to sets of low information content, in: K. Ambos-Spies, S. Homer, U. Schöning (Eds.), Complexity theory, Cambridge University Press, Cambridge, 1993, pp. 1–45.
[4] V. Arvind, J. Köbler, U. Schöning, R. Schuler, If NP has polynomial-size circuits, then MA=AM, Theoretical Computer Science 137 (2) (1995) 279–282.
[5] L. Babai, L. Fortnow, Arithmetization: a new method in structural complexity theory, Computational Complexity 1 (1) (1991) 41–66.
[6] J. Balcázar, Self-reducibility, Journal of Computer and System Sciences 41 (3) (1990) 367–388.
[7] J. Balcázar, R. Book, T. Long, U. Schöning, A. Selman, Sparse oracles and uniform complexity classes, in: Proceedings of the 25th IEEE Symposium on Foundations of Computer Science, 1984, pp. 308–313.
[8] J. Balcázar, R. Book, U. Schöning, The polynomial-time hierarchy and sparse oracles, Journal of the ACM 33 (3) (1986) 603–617.

[9] J. Balcázar, J. Díaz, J. Gabarró, in: Structural Complexity I, second ed., EATCS Texts in Theoretical Computer Science, Springer-Verlag, Berlin, 1995.
[10] J. Balcázar, U. Schöning, Logarithmic advice classes, Theoretical Computer Science 99 (2) (1992) 279–290.
[11] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, C. Tamon, Oracles and queries that are sufficient for exact learning, Journal of Computer and System Sciences 52 (3) (1996) 421–433.
[12] H. Buhrman, J. Kadin, T. Thierauf, Functions computable with nonadaptive queries to NP, Theory of Computing Systems 31 (1) (1998) 77–92.
[13] J. Cai, $S_2^p \subseteq ZPP^{NP}$, in: Proceedings of the 42nd IEEE symposium on foundations of computer science, IEEE Computer Society Press, Silver Spring, MD, 2001, pp. 620–629.
[14] J. Cai, V. Chakaravarthy, L. Hemaspaandra, M. Ogihara, Competing provers yield improved Karp–Lipton collapse results, in: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, vol. 2607, Springer-Verlag, Berlin, 2003, pp. 535–546.
[15] J. Cai, L. Hemaspaandra, G. Wechsung, Robust reductions, Theory of Computing Systems 32 (6) (1999) 625–647.
[16] R. Canetti, More on BPP and the polynomial-time hierarchy, Information Processing Letters 57 (5) (1996) 237–241.
[17] R. Chang, Bounded queries, approximations, and the boolean hierarchy, Information and Computation 169 (2) (2001) 129–159.
[18] R. Gavaldà, J. Balcázar, Strong and robustly strong polynomial time reducibilities to sparse sets, Theoretical Computer Science 88 (1) (1991) 1–14.
[19] Y. Gurevich, Algebras of feasible functions, in: Proceedings of the 24th IEEE symposium on foundations of computer science, IEEE Computer Society Press, Silver Spring, MD, 1983, pp. 210–214.
[20] J. Hartmanis, L. Hemachandra, Complexity classes without machines: On complete languages for UP, Theoretical Computer Science 58 (1–3) (1988) 129–142.
[21] J. Hartmanis, N. Immerman, On complete problems for NP$\bigcap$coNP, in: Proceedings of the 12th International Colloquium on Automata, Languages, and Programming, Lecture Notes in Computer Science, vol. 194, Springer-Verlag, Berlin, 1985, pp. 250–259.
[22] L. Hemaspaandra, A. Hoene, A. Naik, M. Ogiwara, A. Selman, T. Thierauf, J. Wang, Nondeterministically selective sets, International Journal of Foundations of Computer Science 6 (4) (1995) 403–416.
[23] L. Hemaspaandra, S. Jain, N. Vereshchagin, Banishing robust Turing completeness, International Journal of Foundations of Computer Science 4 (3) (1993) 245–265.
[24] L. Hemaspaandra, A. Naik, M. Ogihara, A. Selman, Computing solutions uniquely collapses the polynomial hierarchy, SIAM Journal on Computing 25 (4) (1996) 697–708.
[25] L. Hemaspaandra, M. Ogihara, The complexity theory companion, Springer-Verlag, Berlin, 2002.
[26] J. Hopcroft, Recent directions in algorithmic research, in: Proceedings 5th GI Conference on Theoretical Computer Science, Lecture Notes in Computer Science, vol. 104, Springer-Verlag, Berlin, 1981, pp. 123–134.
[27] N. Immerman, S. Mahaney, Relativizing relativized computations, Theoetical Computer Science 68 (3) (1989) 267–276.
[28] J. Kämper, Non-uniform proof systems: A new framework to describe non-uniform and probabilistic complexity classes, Theoretical Computer Science 85 (2) (1991) 305–331.
[29] R. Karp, R. Lipton, Some connections between nonuniform and uniform complexity classes, in: Proceedings of the 12th ACM symposium on theory of computing, ACM Press, New York, 1980, pp. 302–309, an extended version has also appeared as: Turing machines that take advice, L'Enseignement Mathématique, second series, vol. 28, 1982, pp. 191–209.
[30] K. Ko, On self-reducibility and weak P-selectivity, Journal of Computer and System Sciences 26 (1983) 209–221.
[31] J. Köbler, Locating P/poly optimally in the extended low hierarchy, Theoretical Computer Science 134 (2) (1994) 263–285.
[32] J. Köbler, On the structure of low sets, in: Proceedings of the 10th structure in complexity theory conference, IEEE Computer Society Press, Silver Spring, MD, 1995, pp. 246–261.
[33] J. Köbler, O. Watanabe, New collapse consequences of NP having small circuits, SIAM Journal on Computing 28 (1) (1998) 311–324.
[34] T. Long, Strong nondeterministic polynomial-time reducibilities, Theoretical Computer Science 21 (1982) 1–25.
[35] T. Long, A. Selman, Relativizing complexity classes with sparse oracles, Journal of the ACM 33 (3) (1986) 618–627.

[36] C. Lund, L. Fortnow, H. Karloff, N. Nisan, Algebraic methods for interactive proof systems, Journal of the ACM 39 (4) (1992) 859–868.
[37] A. Meyer, M. Paterson, With what frequency are apparently intractable problems difficult? Tech. Rep. MIT/LCS/TM-126, Laboratory for Computer Science, MIT, Cambridge, MA, 1979.
[38] C. Papadimitriou, Computational complexity, Addison-Wesley, Reading, MA, 1994.
[39] D. Ranjan, S. Chari, P. Rohatgi, Improving known solutions is hard, Computational Complexity 3 (2) (1993) 168–185.
[40] A. Russell, R. Sundaram, Symmetric alternation captures BPP, Computational Complexity 7 (2) (1998) 152–162.
[41] U. Schöning, A low and a high hierarchy within NP, Journal of Computer and System Sciences 27 (1) (1983) 14–28.
[42] U. Schöning, in: Complexity and Structure, Lecture Notes in Computer Science, vol. 211, Springer-Verlag, Berlin, 1986.
[43] A. Selman, Polynomial time enumeration reducibility, SIAM Journal on Computing 7 (4) (1978) 440–457.
[44] A. Selman, A taxonomy of complexity classes of functions, Journal of Computer and System Sciences 48 (2) (1994) 357–381.
[45] J. Simon, On some central problems in computational complexity, Ph.D. thesis, Cornell University, Ithaca, NY, available as Cornell Department of Computer Science Technical Report TR75-224 (1975).
[46] S. Toda, PP is as hard as the polynomial hierarchy, SIAM Journal on Computing 20 (5) (1991) 865–877.
[47] S. Toda, M. Ogihara, Counting classes are at least as hard as the polynomial-time hierarchy, SIAM Journal on Computing 21 (2) (1992) 316–328.
[48] J. Torán, Complexity classes defined by counting quantifiers, Journal of the ACM 38 (3) (1991) 753–774.
[49] V. Variyam, A note on NP∩coNP/poly (note: the author uses NP∩coNP/poly to denote what in the present paper is denoted (NP∩coNP)/poly), Tech. Rep. RS-00-19, BRICS, Aarhus, Denmark, 2000.
[50] N. Vinodchandran, $AM_{exp} \not\subseteq (NP \cap coNP)/poly$, Information Processing Letters, 89(1) (2004) 43–47.
[51] K. Wagner, The complexity of combinatorial problems with succinct input representations, Acta Informatica 23 (3) (1986) 325–356.
[52] K. Wagner, Bounded query computation, in: Proceedings of the 3rd structure in complexity theory conference, IEEE Computer Society Press, Silver Spring, MD, 1988, pp. 260–277.
[53] C. Yap, Some consequences of non-uniform conditions on uniform classes, Theoretical Computer Science 26 (3) (1983) 287–300.
[54] S. Zachos, Robustness of probabilistic complexity classes under definitional perturbations, Information and Computation 54 (3) (1982) 143–154.